Rancangan Model Algoritma Pohlig-Hellman dengan Menggunakan *Multiple-key* Berdasarkan Algoritma RSA *Multiple-key*

Allwin Simarmata
Fakultas Teknologi dan Ilmu Komputer
Universitas Prima Indonesia
Medan, Indonesia
allwin.simarmata@yahoo.com

***Abstrak*—Perkembangan teknologi informasi menuntut keamanan data yang tangguh. Dalam hal keamanan data kriptografi juga berkembang dengan cepat. Banyak algoritma baru yang dibuat dan dikembangkan dengan harapan dapat memberi keamanan data yang lebih baik. Algoritma Pohlig-Hellman dan Algoritma RSA merupakan bagian dari kriptografi dari kelompok algoritma asimetris yang identik dengan pembangkitan bilangan prima sebagai kuncinya.**

Algoritma RSA sudah lebih baik dibandingkan algoritma Pohlig-Hellman yang sudah ada sebelumnya. Hal ini karena Algoritma RSA menggunakan dua bilangan prima sebagai kunci sedangkan algoritma Pohlig-Hellman hanya menggunakan satu kunci saja.

Untuk mengembangkan Algoritma Pohlig-Hellman maka penulis merancang model Algoritma Pohlig-Hellman dengan menggunakan *multiple-key* untuk pengembangan algoritma Pohlig-Hellman sebelumnya.

Rancangan model diperoleh dengan meneliti kekuatan dan kelemahan yang ada pada algoritma Pohlig-Hellman dengan cara membandingkannya dengan algoritma RSA dan algoritma RSA *Multiple-key*. Kekuatan dan kelemahan dari algoritma diukur berdasarkan kecepatan proses dan tingkat kesulitan algoritma untuk ditembus oleh pihak lain.

Kesamaan konsep dalam menggunakan kunci bilangan prima menjadi dasar penulis untuk pengembangan algoritma ini. Teorema Fermat digunakan dalam pembangkitan bilangan prima yang digunakan sebagai kunci pada algoritma. Berdasarkan kelebihan dan kekurangan tersebut selanjutnya didapat rancangan pengembangan algoritma Pohlig-Hellman dengan konsep *multiple-key*. Model ini tentunya lebih baik dibandingkan dengan algoritma Pohlig-Helman sebelumnya karena dengan penambahan kunci berlapis menjadikan algoritma lebih sulit untuk ditembus pada proses enkripsi dan dekripsi.

Kata kunci—Pohlig-Hellman; RSA; Multiple-ke; Teorema Fermat; Kriptografi

I. PENDAHULUAN

Keamanan dan kerahasiaan pesan, data atau informasi menjadi permasalahan yang sangat penting dalam pertukaran atau pengiriman informasi seperti informasi perbankan, pesan rahasia, data rahasia perkantoran atau pertukaran data lainnya. Hal ini sangat penting supaya pesan, data atau informasi tersebut tidak jatuh kepada pihak yang tidak berkepentingan untuk mengetahuinya (*Bishop*, 2010). Berbagai ancaman yang mungkin terjadi berupa interupsi, penyadapan, maupun modifikasi pada informasi yang akan dikirim. Kriptografi merupakan salah satu cara yang dapat digunakan untuk mengatasi permasalahan ini.

Kriptografi merupakan ilmu dan seni untuk menjaga keamanan pesan ketika dikirim dari sebuah sumber informasi ke suatu tujuan pengiriman informasi (*Ariyus*, 2008). Matematika adalah landasan dasar dalam ilmu kriptografi. Sebagian besar proses dalam kriptografi menggunakan perhitungan matematis baik dalam perhitungan kunci rahasia ataupun dalam perhitungan kode rahasia yang akan digunakan.

Berdasarkan penggunaan kuncinya algoritma kriptografi dikelompokkan atas dua bagian besar yaitu algoritma simetris dan algoritma asimetris. Algoritma simetris menggunakan kunci yang sama untuk melakukan enkripsi dan dekripsi. Beberapa contoh algoritma simetris antara lain *Data Encryption Standard* (DES), RC2, RC4, RC5, RC6, *International Data Encryption Algorithm* (IDEA), *Advanced Encryption Standard* (AES), *One Time Pad* (OTP), A5 dan sebagainya). Algoritma asimetri menggunakan kunci yang berbeda untuk enkripsi dan dekripsi. Beberapa contoh algoritma asimetris antara lain *Digital Signature Algorithm* (DSA), RSA, Diffie-Hellman, Pohlig-Hellman, *Elliptic Curve Crypthography* (ECC), Kriptografi Quantum dan sebagainya. (Ariyus, 2008).

Pada dasarnya cara kerja algoritma Pohlig-Hellman lebih sederhana dibandingkan dengan algoritma RSA karena hanya menggunakan satu bilangan prima sebagai kunci privat, sedangkan untuk algoritma RSA menggunakan dua bilangan prima untuk pembangkitan kunci publik. Namun dengan penggunaan dua bilangan tersebut menjadikan RSA lebih rumit untuk ditembus keamanan datanya.

Dalam teori bilangan, algoritma Pohlig Hellman bertujuan khusus untuk untuk menghitung logaritma diskrit dalam kelompok multiplikatif yang diurutkan berupa bilangan bulat (Thiong-Ly, 1993). Algoritma ini ditemukan oleh Roland Silver, namun pertama kali diterbitkan oleh Stephen Pohlig dan Martin Hellman.

Algoritma RSA merupakan bagian dari kriptografi asimetris yang dijabarkan pertama sekali oleh tiga orang alumni dari *Massachusetts Institute of Technology* yaitu Ron Rivest, Adi Shamir dan Len Adleman pada tahun 1977. Huruf RSA tersebut berasal dari inisial nama mereka yaitu Rivest, Shamir dan Adleman. Algoritma RSA adalah algoritma yang menggunakan enkripsi kunci publik. Algoritma RSA banyak digunakan dan dipercaya dalam mengamankan data atau informasi menggunakan kunci dengan bilangan yang cukup besar. Algoritma tersebut dipatenkan oleh *Massachusetts Institute of Technology* pada tahun 1983. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci pribadi. Sejauh ini belum ditemukan algoritma yang lebih baik dari RSA (William, 2005).

Dengan melakukan pembangkitan dua bilangan prima, algoritma RSA membutuhkan waktu yang cukup banyak dalam melakukan prosesnya. Namun algoritma RSA lebih banyak dikembangkan karena konsep yang cukup rumit untuk ditembus. Salah satu pengembangan Algoritma RSA adalah dengan konsep *multiple-key*. *Multiple-key* adalah penambahan kunci-kunci pada proses enkripsi dan dekripsi yang digunakan untuk meningkatkan keamanan data. (Shcneider, 2006)

Konsep *multiple-key* dapat digunakan dalam untuk algoritma asimetris karena dapat meningkatkan kualitas dan kekuatan dari kunci yang sudah ada sebelumnya. Pengembangan dari algoritma Pohlig-Hellman dengan *multiple-key* adalah ide yang dikembangkan dalam tulisan ini. Penerapan *multiple-key* pada algoritma Pohlig-Hellman diharapkan akan dapat meningkatkan kehandalan algoritma dalam keamanan data. Adanya beberapa kesamaan konsep antara algoritma Pohlig-Hellman dan algoritma RSA dalam proses pembangkitan bilangan prima dan proses yang lainnya menjadi dasar untuk membuat rancangan model Algoritma Pohlig-Hellman *multiple-key.*

Pengujian bilangan prima merupakan cara dalam penentuan bilangan prima yang sudah diformula dan akan digunakan dalam pembangkitan kunci publik dalam algoritma Pohlig-Hellman dan RSA (Teske, 1999). Terdapat beberapa teori tentang pengujian bilangan prima seperti Algoritma *Primality Proving*, Pengujian Bilangan Prima Lucas-Lehmer dalam Bilangan Mersenne, Teorema Pocklington, Teorema Proth, Pengujian Prima Pepin, *Fermat's Little Theorem*, *Primes is in P*, *Sophie Germain's Prime Density Conjecture* dan beberapa pengujian bilangan prima lainnya (Mollin, 2007). Pengujian bilangan prima pada setiap teorema mempunyai cara kerja yang berbeda dan beberapa diantaranya ada yang sederhana dan cukup membuktikan keprimaan suatu bilangan.

Dalam tulisan ini menggunakan pembangkitan kunci bilangan prima teorema Fermat. Pada dasarnya teorema Fermat mampu melakukan pengujian bilangan prima dengan pengujian yang baik. Dengan syarat rumus $a^{p-1} \bmod = 1$, dimana p merupakan bilangan yang akan diuji keprimaannya maka akan dilakukan pengujian berulang untuk menguji setiap bilangan apakah prima atau tidak namun dengan syarat bahwa nilai $1 \leq a < p$.

II. PERBANDINGAN ALGORITMA

Untuk mendapatkan gambaran kelebihan dan kekurangan dari algoritma Pohlig-Hellman sebelum dan sesudah penambahan *multiple-key* tersebut dilakukan perban-dingan algoritma-algoritma yang dibahas sebelumnya. Algoritma yang dibandingkan yaitu :

1. Perbandingan algoritma Pohlig-Hellman dengan algoritma RSA;
2. Perbandingan algoritma RSA dengan algoritma RSA *multiple-key*;
3. Perbandingan algoritma RSA *multiple-key* dengan model algoritma Pohlig-Hellman *multiple-key*;
4. Perbandingan algoritma Pohlig-Hellman dengan model algoritma Pohlig-Hellman *multiple-key*.

Terlebih dahulu akan dibahas perbandingan antara Pohlig-Hellman dengan algoritma RSA pada Tabel 1.1

TABLE I. PERBANDINGAN ALGORITMA POHLIG-HELLMAN DAN ALGORITMA RSA

Proses	ALGORITMA POHLIG-HELLMAN	ALGORITMA RSA
1.	Pembangkitan satu bilangan prima p	Pembangkitan dua bilangan prima p dan q, $p \neq q$
2.	-	Perhitungan nilai n $n = p * q$

3.	Perhitungan nilai totient *totient (p) = p-1*	Perhitungan nilai totient *totient (n) = (p-1)*(q-1)*
4.	Perhitungan nilai kunci enkripsi *e* Syarat : *1<e<totient (p)* GCD *(totient(p),e)=1*	Perhitungan nilai kunci dekripsi *e* Syarat : *1<e<totient (n)* GCD *(totient (n),e)) = 1*
5.	Perhitungan nilai kunci dekripsi *d* Syarat : $d = e^{-1} \bmod totient\ (p)$	Perhitungan nilai kunci dekripsi *d* Syarat : $d = e^{-1} \bmod totient\ (n)$
6.	Enkripsi $C = M^e \bmod p$ Dekripsi $M = C^d \bmod p$	Enkripsi $C = M^e \bmod n$ Dekripsi $M = C^d \bmod n$

Dari tabel 1. terlihat perbedaan antara algoritma Pohlig-Hellman dan algoritma RSA adalah pada pembangkitan kuncinya dimana algoritma Pohlig-Hellman menggunakan satu kunci sedang algoritma RSA menggunakan dua kunci. Hal ini mempengaruhi perhitungan nilai *totient* yang didapat, dimana pada algoritma Pohlig-Hellman didapat dari hasil pengurangan nilai *p* dikurang satu, sedangkan pada algoritma RSA didapat dari perhitungan perkalian kedua kunci setelah dikurang satu.

Algoritma RSA *multiple-key* memiliki beberapa perbedaan dengan algoritma RSA yang sudah ada sebelumnya. Untuk melihat perbedaan tersebut dapat dilihat pada Tabel 1.2. berikut ini.

TABLE II. PERBANDINGAN ALGORITMA RSA DAN RSA *MULTIPLE-KEY*

Proses	**ALGORITMA RSA**	**ALGORITMA RSA *Multiple-key***
1.	Pembangkitan 2 Bilangan Prima *p* dan *q* , $p \neq q$	Pembangkitan 2 Bilangan Prima *p* dan *q* , $p \neq q$
2.	Perhitungan nilai n $n = p * q$	Perhitungan nilai n $n = p * q$
3.	Perhitungan nilai totient *totient (n) = (p-1)*(q-1)*	Perhitungan nilai totient *totient (n) = (p-1)*(q-1)*
3.a.	-	Perhitungan *multiple-key* enkripsi *(Ke)* *Syarat :* K_{e1} bil.ganjil & GCD (K*e1*,tot(*n*))= 1 K_{e2} bil.ganjil & GCD (K*e1**K*e2*, tot(*n*))= 1 . . K_{ex} bil. ganjil & GCD (K*e1**..K*ex*, tot(*n*))= 1
3.b	Perhitungan nilai kunci enkripsi *e*, syarat : *1<e<totient (n) & GCD (totient (n) , e)) = 1*	Perhitungan nilai kunci enkripsi *e* *e = (Ke1 * Ke2 * ……. Kex)*
4.a.	-	Perhitungan multiple-key dekripsi (Kd) Syarat : K_{d1} bil.ganjil & GCD (K_{d1},tot(n))= 1 K_{d2} bil.ganjil & GCD (K_{d1}*K_{e2}, tot(n))= 1 . . K_{dx} bil. ganjil & GCD (K_{d1}*..K_{ex}, tot(n))= 1
4.b.	Perhitungan nilai kunci dekripsi *d* Syarat : $d = e^{-1} \bmod totient\ (n)$	Perhitungan nilai kunci dekripsi *d* Syarat : $d = K_{d1} * K_{d2} \ldots\ldots K_{dx}$
5.	Enkripsi $C = M^e \bmod n$ Dekripsi $M = C^d \bmod n$	Enkripsi $C = M^e \bmod n$ Dekripsi $M = C^d \bmod n$

Perbandingan algoritma RSA dan algoritma RSA *multiple-key* memperlihatkan konsep yang sama pada proses pembangkitan kunci, perhitungan nilai *n*, perhitungan nilai totient, serta proses enkripsi dan dekripsinya. Perbedaannya terdapat pada algoritma RSA *multiple-key* yaitu dengan adanya proses perhitungan kunci tambahan (*multiple-key*) sehingga perhitungan nilai kunci enkripsi dan kunci dekripsi didapati dari hasil perkalian kunci tambahan.

Dengan konsep penambahan kunci algoritma akan melakukan perhitungan kunci enkripsi dan dekripsi dengan nilai yang lebih besar dari algoritma RSA sebelumnya yang kuncinya antara *1<e<totient (n)* dan nilai *GCD (totient (n),e))*=1.

III. PERANCANGAN ALGORITMA

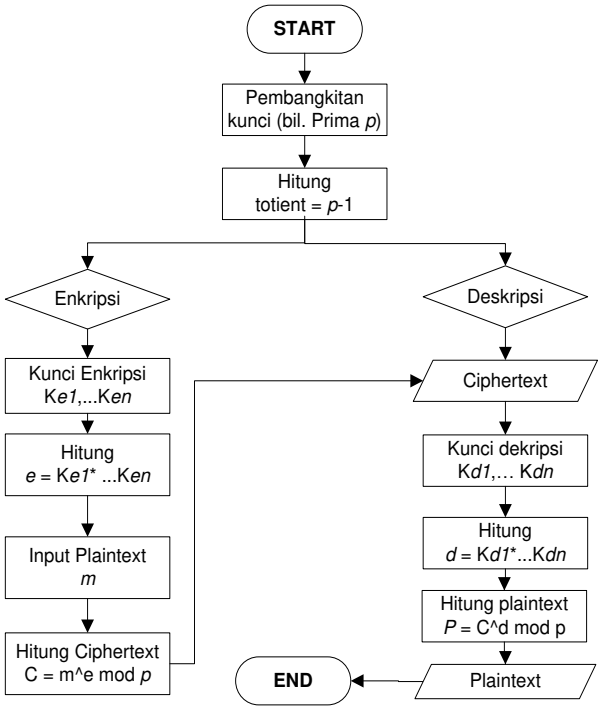
Terdapat dua hal penting yang dilakukan dalam sistem yang dirancang yaitu enkripsi dan dekripsi. Dalam melakukan enkripsi data, terlebih dahulu menentukan kunci berupa bilangan prima yang didapat berdasarkan Teorema Fermat dan dihasilkan secara acak dari bilangan prima yang disepakati. Dengan dilakukan secara acak maka tingkat keamanan kunci publik juga akan lebih tinggi karena hanya diketahui oleh pengirim dan selanjutnya diketahui penerima melalui pengirim. Perhitungan nilai totient dilakukan dengan pengurangan nilai *p* dengan 1.

Nilai totient diperlukan dalam proses enkripsi dan dekripsi. Dalam proses enkripsi selanjutnya akan

ditentukan nilai kunci tambahan (*multiple-key*) yang didapat berdasarkan dari GCD antara nilai kunci yang mungkin dengan nilai totient harus bernilai 1. Kunci tambahan yang dihasilkan nantinya akan digunakan dalam perhitungan nilai kunci enkripsi *e*. Nilai *e* didapat dengan perkalian setiap kunci tambahan yang digunakan.

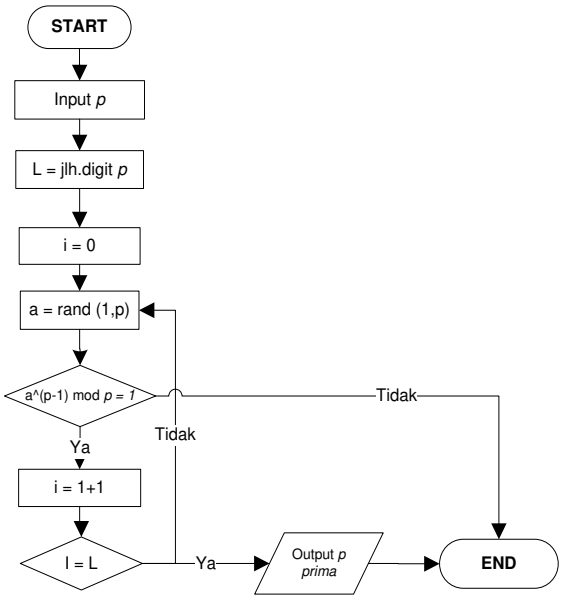
Jika semua nilai telah terpenuhi maka dapat dihitung *ciphertext* dari *plaintext* yang diinput. Demikian halnya untuk melakukan proses dekripsi dapat dilakukan berdasarkan *ciphertext* yang didapat. Setiap *plaintext* yang diinput satu persatu akan dikodekan dalambentuk *ciphertext* dengan perhitungan pada proses enkripsi. Kode tersebut dihitung berdasarkan nilai plaintext pada kode ASCII.

Dengan menentukan kunci tambahan dekripsi sebanyak *n*-buah yang didapat secara acak dari GCD antara kunci yang memungkinkan dengan nilai totient harus bernilai 1. Nilai dekripsi *d* dapat dihitung dengan perkalian setiap kunci tambahan. Dengan nilai *d* yang didapat maka *plaintext* dapat dihitung kembali berdasarkan rumus dekripsi.



Gambar 1. Rancangan Algoritma Pohlig-Hellman *Multiple-key*

Dari rancangan Algoritma Pohlig-Hellman Multiple-key, maka *multiple-key* diterapkan dalam algoritma Pohlig-Hellman dengan menyisipkan kunci tambahan dalam algoritma. Kunci tambahan tersebut ditempatkan setelah mendapatkan nilai totient, sehingga untuk konsep kunci tambahan mengadopsi kunci tambahan yang ada pada algoritma RSA *multiple-key*.



Gambar 2. *Flowchart* Teorema Fermat

Pada pembangkitan kunci bilangan prima dilakukan seperti pada teorema Fermat, dengan mengeksekusi setiap bilangan yang sudah ditentukan batasannya pada program. Selanjutnya program akan secara acak menentukan bilangan prima yang akan digunakan.

Tabel 4.1. memperlihatkan perbandingan algoritma RSA *multiple-key* dan model algoritma Pohlig-Hellman menggunakan *multiple-key*.

TABLE III. PERBANDINGAN ALGORITMA RSA MULTIPLE-KEY DAN MODEL ALGORITMA POHLIG-HELLMAN MULTIPLE-KEY

Proses	**ALGORITMA RSA *Multiple-key***	**ALGORITMA POHLIG-HELLMAN *Multiple-key***
1.	Pembangkitan dua bilangan prima p dan q , $p \neq q$	Pembangkitan satu bilangan prima
2.	Perhitungan nilai n $n = p * q$	-
3.	Perhitungan nilai totient *totient* $(n) = (p-1)*(q-1)$	Perhitungan nilai totient *totient* $(p) = p-1$

Proses	ALGORITMA RSA *Multiple-key*	ALGORITMA POHLIG-HELLMAN *Multiple-key*
3.a.	Perhitungan *multiple-key* enkripsi *(Ke)* *Syarat :* K_{e1} bil.ganjil &GCD(K_{e1},tot(n))= 1 K_{e2} bil.ganjil&GCD (K_{e1}*K_{e2},tot(n) = 1 . . K_{ex} bil.ganjil & GCD (K_{e1}*..K_{ex}, tot(n)= 1	Perhitungan *multiple-key* enkripsi (Ke) Syarat : K_{e1} bil.ganjil&GCD (K_{e1},tot(p)) =1 K_{e2} bil.ganjil&GCD (K_{e1}*K_{e2},tot(p) = 1 . . K_{ex} bil.ganjil & GCD (K_{e1}*..K_{ex}, tot(p))= 1
3.b	Perhitungan nilai kunci dekripsi *e* $e = (K_{e1} * K_{e2} * \ldots\ldots K_{ex})$	Perhitungan nilai kunci enkripsi *e* $e = (K_{e1} * K_{e2} * \ldots\ldots K_{ex})$
4.a.	Perhitungan multiple-key dekripsi (K_d) Syarat: K_{d1} bil.ganjil & GCD (K_{d1},tot(n) =1 K_{d2} bil.ganjil&GCD (K_{d1}*K_{d2}, tot(n) = 1 . . K_{dx} bil. ganjil & GCD (K_{d1}*..K_{ex}, tot(n)) = 1	Perhitungan multiple-key dekripsi (K_d) Syarat : K_{d1} bil.ganjil & GCD (K_{d1},tot(p)) =1 K_{d2} bil.ganjil & GCD (K_{d1}*K_{d2}, tot(p))= 1 . . K_{dx} bil. ganjil & GCD (K_{d1}*..K_{ex}, tot(p))= 1
4.b.	Perhitungan nilai kunci dekripsi *d* Syarat : $d = K_{d1} * K_{d2} \ldots\ldots K_{dx}$	Perhitungan nilai kunci dekripsi *d* Syarat : $d = K_{d1} * K_{d2} \ldots\ldots K_{dx}$
5.	Enkripsi $C = M^e \bmod n$ Dekripsi $M = C^d \bmod n$	Enkripsi $C = M^e \bmod p$ Dekripsi $M = C^d \bmod p$

Dari perbandingan kedua algoritma didapat persamaan dalam perhitungan kunci *multiple-key,* namun tetap mengikuti cara kerja algoritma dasarnya masing-masing. Setelah mendapatkan perbandingan dari algoritma-algoritma diatas terlihat jelas bahwa penggunaan *multiple-key* pada algoritma Pohlig-Hellman meningkatkan kualitas kerahasiaan informasi karena dijaga oleh beberapa lapisan kunci yang dapat mempersulit pihak lain untuk menembus kunci-kunci tersebut.

Berdasarkan perbandingan-perbandingan diatas maka dapat dilihat perbandingan antara algoritma Pohlig-Hellman sebelum dan sesudah menggunakan *multiple-key*. Pada tabel 4.2. diperlihatkan perbandingan antara kedua algoritma tersebut.

TABLE IV. PERBANDINGAN MODEL ALGORITMA POHLIG-HELLMAN MULTIPLE-KEY DAN RSA MULTIPLE-KEY

Proses	ALGORITMA POHLIG-HELLMAN	ALGORITMA POHLIG-HELLMAN *Multiple-key*
1.	Pembangkitan satu bilangan prima *p*	Pembangkitan satu bilangan prima
2.	-	-
3.	Perhitungan nilai totient *totient (p) = p-1*	Perhitungan nilai totient *totient (p) = p-1*
3.a.	-	Perhitungan *multiple-key* enkripsi (Ke) Syarat : K_{e1} bil.ganjil&GCD (K_{e1},tot(p)) =1 K_{e2} bil.ganjil&GCD (K_{e1}*K_{e2},tot(p) = 1 . . K_{ex} bil.ganjil & GCD (K_{e1}*..K_{ex}, tot(p))= 1
3.b	Perhitungan nilai kunci enkripsi *e* Syarat : $1 < e < totient (p)$ GCD *(totient(p), e] = 1*	Perhitungan nilai kunci enkripsi *e* $e = (K_{e1} * K_{e2} * \ldots\ldots K_{ex})$
4.a.	-	Perhitungan multiple-key dekripsi (K_d) Syarat : K_{d1} bil.ganjil & GCD (K_{d1},tot(p)) =1 K_{d2} bil.ganjil & GCD (K_{d1}*K_{d2}, tot(p))= 1 . . K_{dx} bil. ganjil & GCD (K_{d1}*..K_{ex}, tot(p))= 1
4.b.	Perhitungan nilai kunci dekripsi *d* Syarat : $d = e^{-1} \bmod totient (p)$	Perhitungan nilai kunci dekripsi *d* Syarat : $d = K_{d1} * K_{d2} \ldots\ldots K_{dx}$
5.	Enkripsi $C = M^e \bmod p$ Dekripsi $M = C^d \bmod p$	Enkripsi $C = M^e \bmod p$ Dekripsi $M = C^d \bmod p$

Dari tabel 4.2. terlihat perbedaan antara algoritma Pohlig-Hellman sebelum dan sesudah dikembangkan dengan konsep *multiple-key*. Konsep *multiple-key* menjadi perbedaan utama dari kedua algoritma. Dengan *multiple-key* akan menghasilkan nilai kunci enkripsi dan dekripsi yang lebih besar karena didapat dari perkalian kunci-kunci tambahannya. Kunci dengan nilai yang besar pastinya akan melakukan perhitungan yang besar pula untuk proses enkripsi dan dekripsinya.

IV. UJI COBA PROGRAM

Dari hasil pengujian rancangan algoritma yang sudah dibuat, terdapat beberapa perbedaan antara algoritma Pohlig-Hellman menggunakan *multiple-key* dengan algoritma Pohlig-Hellman tanpa multiple-key. Perbedaan tersebut mencakup pada proses pembangkitan kunci privat, penentuan kunci enkripsi dan kunci dekripsi, perhitungan kode ciphertext, kecepatan proses sistem untuk melakukan enkripsi dan dekripsi. Namun algoritma Pohlig Hellman sebelumnya menjadi dasar untuk pengembangan algoritma. Hal ini dibuktikan karena terdapat beberapa kesamaan yang mendasar antara kedua algoritma seperti pembangkitan kunci bilangan prima, perhitungan nilai totient dan proses enkripsi dan dekripsi.

Berdasarkan hasil pengujian dari kedua algoritma, dapat dilihat adanya kelebihan dan kekurangan dari kedua algoritma. Adapun beberapa hasil yang didapat dari pengujian program dapat dilihat pada Tabel 4.5. berikut ini.

TABLE V. PERBANDINGAN PROSES PADA PENGUJIAN PROGRAM

No	Perbandingan Proses	Algoritma Pohlig-Hellman	
		Multiple-key	Tanpa *Multiple-key*
1	Pembangkitan kunci privat	Menggunakan pembangkitkan kunci privat yang sama yaitu bilangan prima yang dihitung teorema Fermat dan bilangan dihasilkan secara acak	
2	Perhitungan nilai totient	Menggunakan perhitungan yang sama yaitu $tot\ (p) = p - 1$	
3	Pembangkitan kunci tambahan (*multiple-key*)	Menggunakan pembangkitan kunci tambahan untuk enkripsi dan dekripsi	Tidak menggunakan kunci tambahan
4	Perhitungan kunci enkripsi	Dilakukan dengan perkalian kunci tambahan (*multiple-key*) enkripsi	Dilakukan dengan perhitungan GCD*(tot (p), e) = 1 dan 1<e<(tot)(p)*
5	Perhitungan kunci dekripsi	Dilakukan dengan perkalian kunci tambahan (*multiple-key*) dekripsi	Dilakukan dengan perhitungan : $d = e^{-1} \bmod p$
6	Perhitungan kode ciphertext	Proses dilakukan lebih lama karena nilai kunci enkripsi lebih besar	Proses dilakukan lebih cepat karena nilai kunci enkripsi kecil
7	Proses dekripsi	Proses dilakukan lebih lama karena nilai kunci dekripsi lebih besar	Proses dilakukan lebih cepat karena nilai kunci dekripsi kecil
8	Kecepatan proses sistem	Proses dilakukan lebih lambat karena melakukan penambahan kunci	Proses dilakukan lebih cepat karena hanya menggunakan satu kunci
9	Kekuatan sistem	Lebih sulit ditembus karena dilapis oleh kunci tambahan	Dibandingkan dengan penambahan *multiple-key* lebih rendah kekuatan karena hanya menggunakan satu kunci.

V. KESIMPULAN

Berdasarkan hasil percobaan diatas, kami mengajukan kesimpulan sebagai berikut:

a. Rancangan model algoritma Pohlig-Hellman *multiple-key* dapat bekerja lebih baik dibandingkan dengan algoritma Pohlig-Hellman sebelumnya karena dengan konsep penambahan kunci (*multiple-key*) menjadikan kerahasiaan informasi berlapis sehingga algoritma lebih sulit untuk ditembus.
b. Konsep *multiple-key* pada *model* algoritma Pohlig-Hellman meningkatkan kerahasiaan informasi dengan kunci yang berlapis sehingga dapat mempersulit pihak lain untuk memecahkan kunci enkripsi maupun kunci dekripsi.
c. Pembangkitan kunci secara acak untuk kunci privat dan kunci tambahan (*multiple-key*) pada model algoritma meningkatkan kualitas kunci yang digunakan karena lebih sulit untuk ditebak nilai dari kunci yang digunakan.
d. Dalam pengujian pembangkitan kunci bilangan prima jika diuji dengan batas maksimum sampai 1.000, sistem terlalu berat untuk menjalankan perhitungan karena membutuhkan *processor* dan bahasa pemrograman yang mampu memproses dengan perhitungan dengan jumlah yang besar.

DAFTAR PUSTAKA

[1] Ariyus, D. 2008 Pengantar Ilmu Kriptografi Teori Analisis dan Implementasi. Penerbit Andi, Yogyakarta.

[2] Bishop, M. 2010 *Introduction to Computer Security*, Pearson Education. Inc.

[3] Chen, R. 2008 *The Pohlig-Hellman Method*, ECC Journal 2008, Department of Computer Science National Chia Tung University, Cryptanalysis Lab., pp.29-38.

[4] Diffie W. & Hellman, M. E. 2003 *New Direction in Cryptography, IEEE Transaction on Information Theory-22*, no.6, pp.644-654.

[5] Hellman, M. E. 1980 *A Cryptanalytic Time – Memory Trade-Off*, IEEE Transaction on Information Theory, Vol. IT-26 No. 4.

[6] Hellman, M. E. 2002 *An Overview of Public Key Cryptography*. IEEE Communications Magazine 50th Anniversary Issue: Landmark 10 Papers, pp.42-49.
[7] Holden, J. 2000 *The Pohlig-Hellman Exponentiation Cipher as a Bridge Between Classical And Modern Cryptography*, Rose-Hulman Institute of Technology.
[8] Menezes, A. J., Paul C. V. O., & Scott A. V. 1996 *Handbook of Applied Cryptography*, CRC Press.
[9] Mollin, R. A. 2007 *An Introduction to Cryptography - 2nd Edition*. Chapman and Hall / CRC
[10] Pohlig S. C., & Hellman M. E., 1978 *An improved algorithm for computing locarithm, over GF(p) and its cryptographic significance*, IEEE Trans. Information Theory, vol. IT-24, pp. 918-924
[11] Schmeh, K. 2003 *Cryptography and Public Key Infrastructure on the Internet*. West Sussex, Jhon Wiley & Sons Ltd.
[12] Schneider, B. 1996 *Aplied Cryptography- 2nd*, John Wiley & Sons, New York
[13] Smart, N. 2004 *Cryptography: An Introduction (3rd Edition)*, University of Bristol.
[14] Stallings, W. 2005 *Cryptography and Network Security Principles and Practices, 4th Edition*, Prentice Hall
[15] Teske, E. 1999 *The Pohlig-Hellman Method Generalized for Group Structure Computation. Symbolic Computation*, University of Waterloo,Canada, pp.521-534